(12) UK Patent Application (19) GB (11) 2 325 761 (13) A

(43) Date of A Publication 02.12.1998

(21) Application No 9806003.1

(22) Date of Filing 23.03.1998

(30) Priority Data
(31) 08828897 (32) 31.03.1997 (33) US

(51) INT CL[6]
G06F 17/30

(52) UK CL (Edition P)
G4A AUDB

(56) Documents Cited
None

(58) Field of Search
UK CL (Edition P) G4A AUDB
INT CL[6] G06F 17/30
Online: WPI,COMPUTER

(71) Applicant(s)
International Business Machines Corporation
(Incorporated in USA - New York)
Armonk, New York 10504, United States of America

(72) Inventor(s)
Howard Justin Glaser

(74) Agent and/or Address for Service
Alan Burrington & Associates
4 Burney Close, Great Bookham, LEATHERHEAD,
Surrey, KT22 9HW, United Kingdom

(54) Abstract Title
A smart guide for the development of customised database extender programs

(57) A smart guide supports the development of internet and intranet applications, including the production of customised database extender programs. Information from a graphic user interface (figure 7) is captured 800 and an intermediate language generated therefrom 802. Source code is then generated 804 from the intermediate language. The source code implements the function of the database extender by interfacing to a generic extender. The interfaced generic extender 806 is configured to adopt the characteristics of the specific extender defined in the intermediate language. The configured generic extender is then executed 810 to access and maintain the relational integrity of the specific extender.

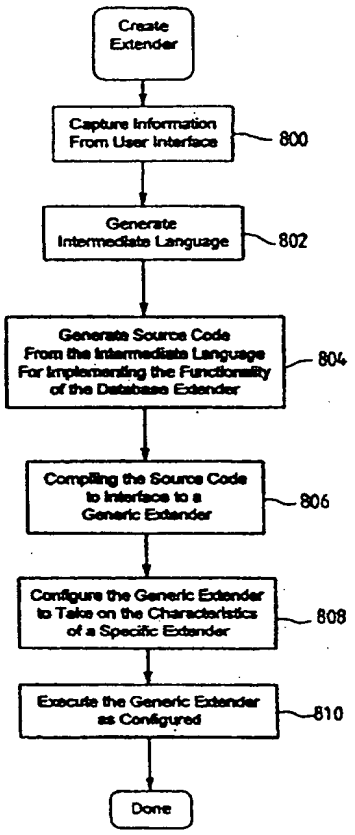

FIG. 8

GB 2 325 761 A

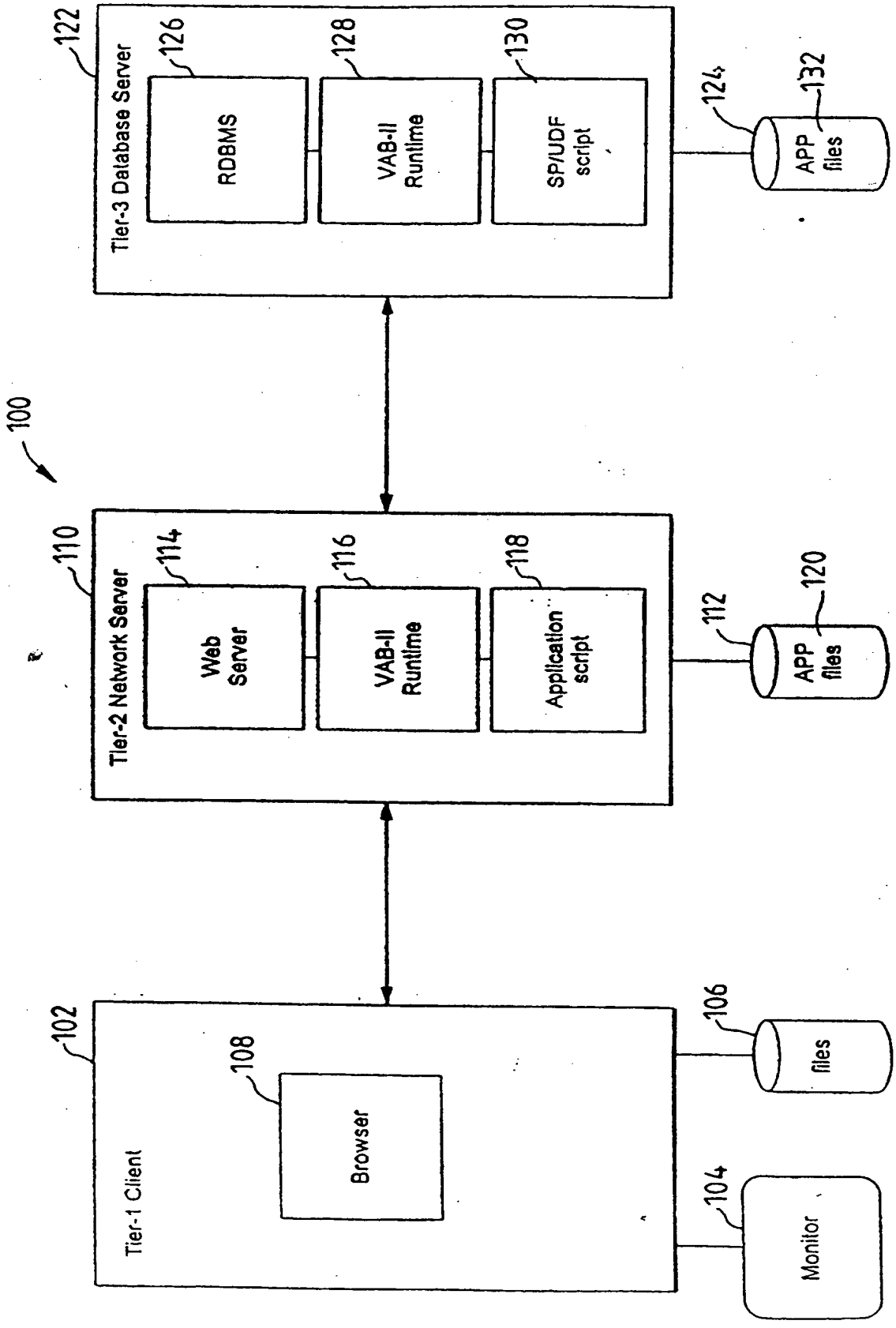
100
Tier-3 Database Server
122
RDBMS
126
VAB-II Runtime
128
SP/UDF script
130
APP files
124
132
Tier-2 Network Server
110
Web Server
114
VAB-II Runtime
116
Application script
118
APP files
112
120
Tier-1 Client
102
Browser
108
files
106
Monitor
104

FIG. 1

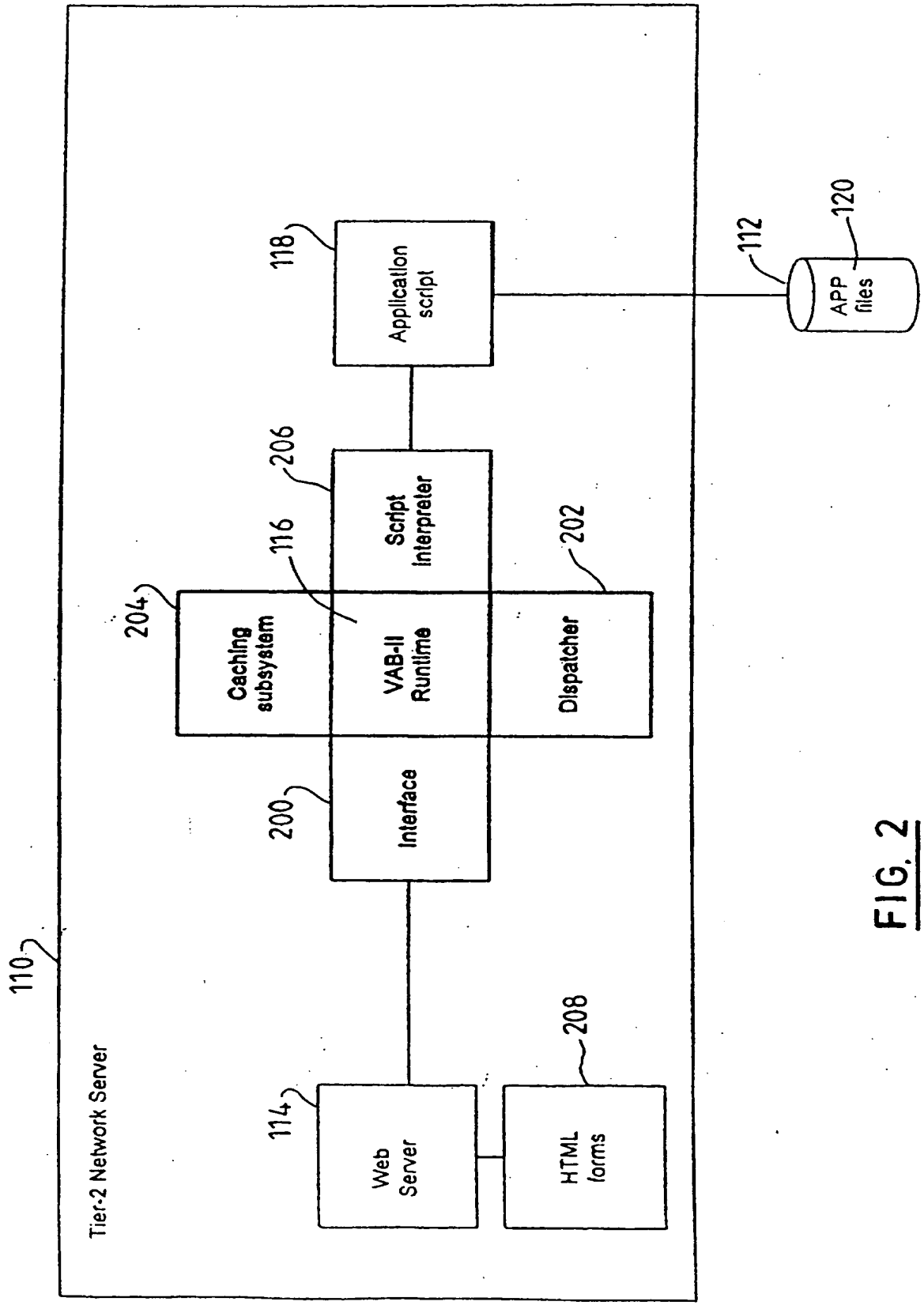
110
Tier-2 Network Server
Web Server
114
HTML forms
208
Interface
200
Caching subsystem
204
VAB-II Runtime
116
Dispatcher
202
Script Interpreter
206
Application script
118
APP files
120
112

FIG. 2

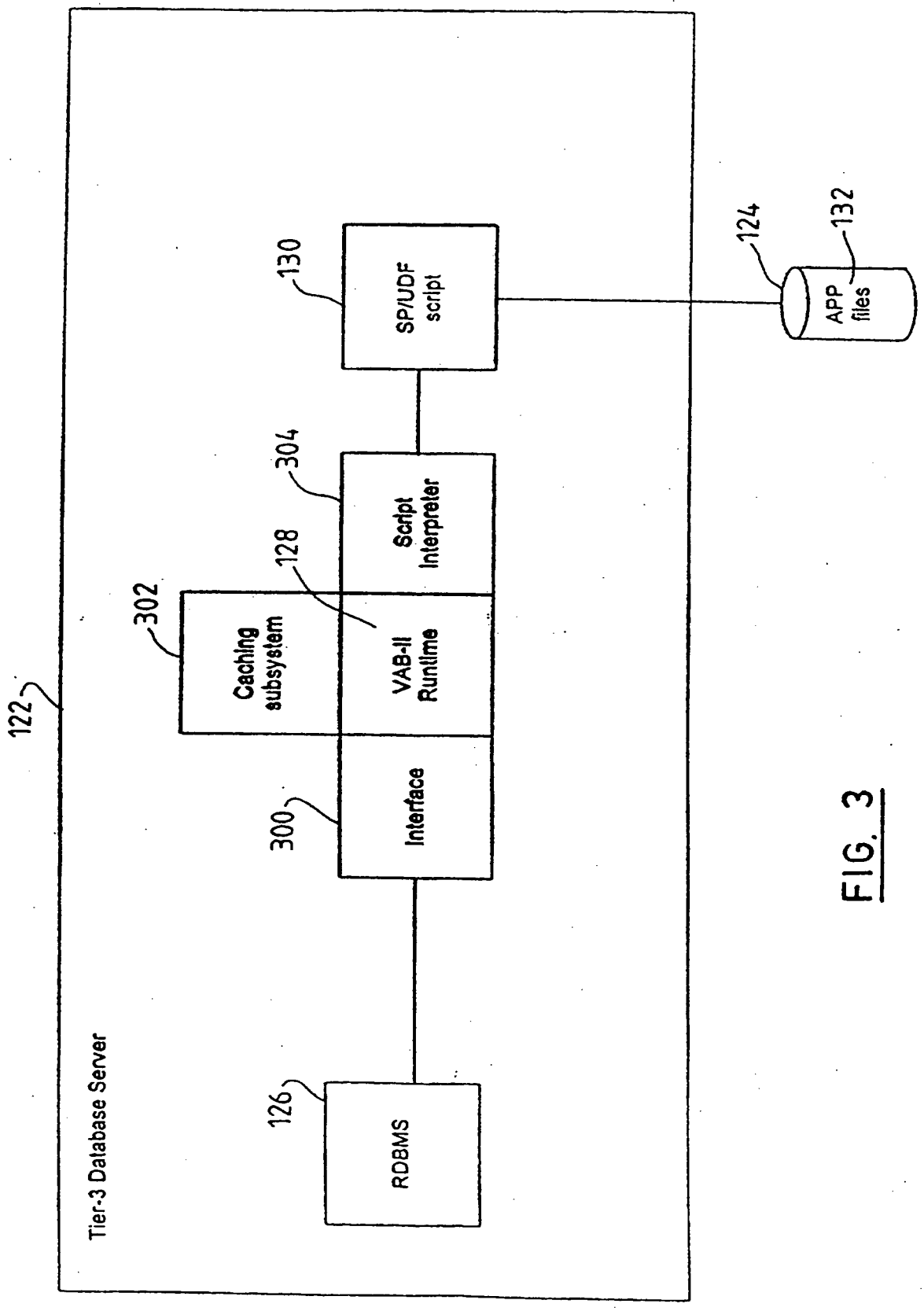
122
Tier-3 Database Server
RDBMS
126
300
Interface
VAB-II Runtime
128
Caching subsystem
302
304
Script Interpreter
SP/UDF script
130
124
APP files
132

FIG. 3

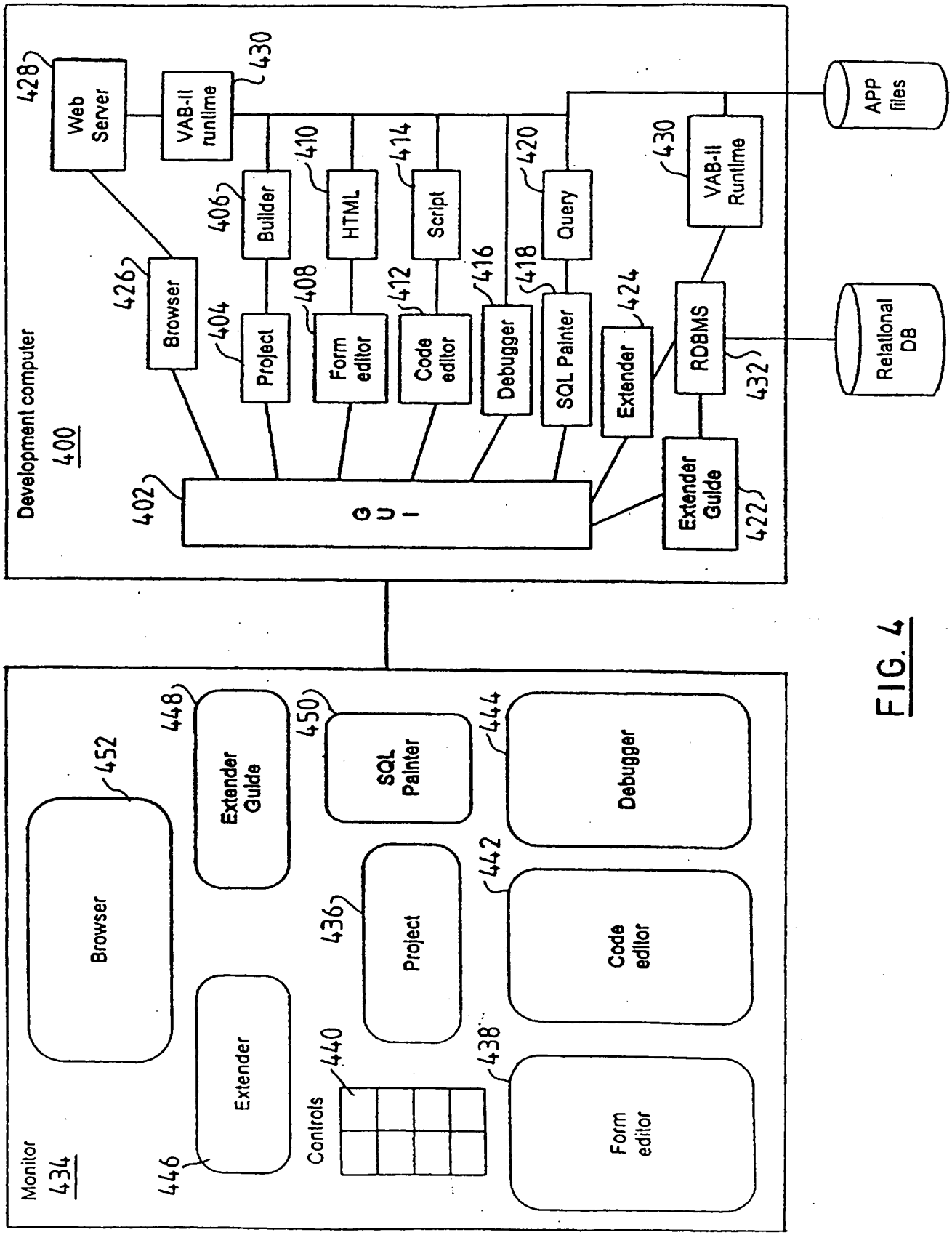
Development computer
400
Web Server
428
VAB-II runtime
430
Builder
406
HTML
410
Script
414
Query
420
VAB-II Runtime
430
APP files
Browser
426
Project
404
Form editor
408
Code editor
412
Debugger
416
SQL Painter
418
Extender
424
RDBMS
432
Relational DB
GUI
402
Extender Guide
422
Monitor
434
Browser
452
Extender Guide
448
SQL Painter
450
Debugger
444
Project
436
Code editor
442
Extender
446
Controls
440
Form editor
438

FIG. 4

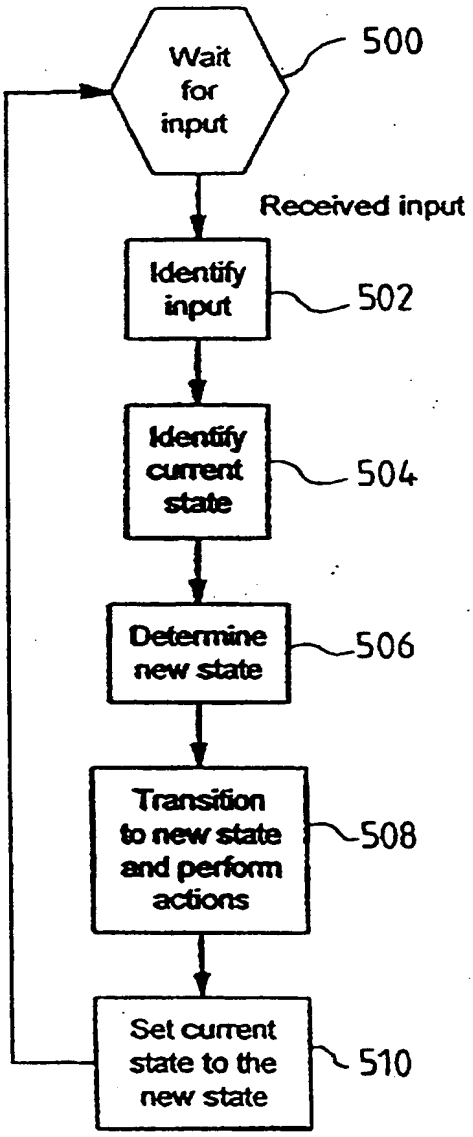
Wait for input
500
Received input
Identify input
502
Identify current state
504
Determine new state
506
Transition to new state and perform actions
508
Set current state to the new state
510

FIG. 5

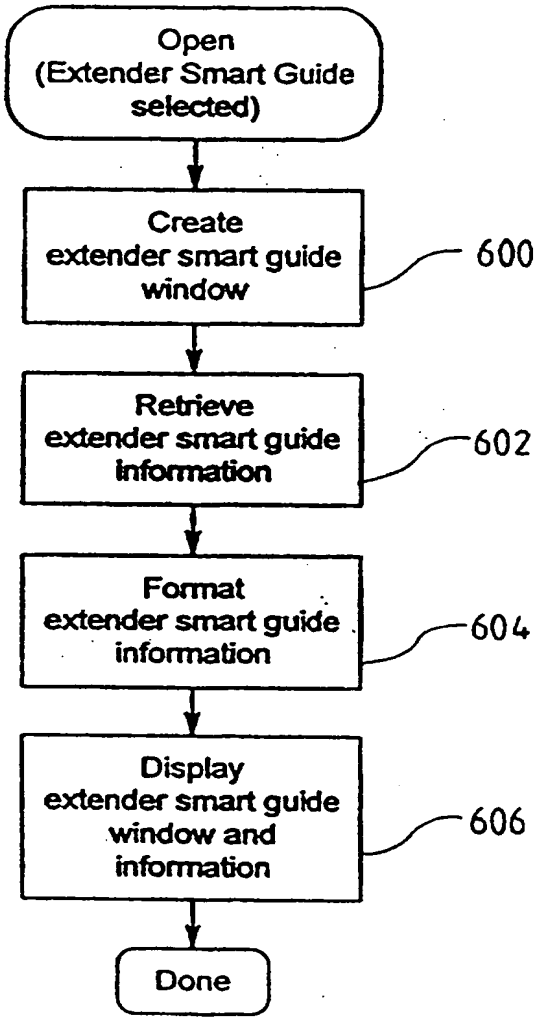
Open
(Extender Smart Guide selected)
Create extender smart guide window
600
Retrieve extender smart guide information
602
Format extender smart guide information
604
Display extender smart guide window and information
606
Done

FIG. 6

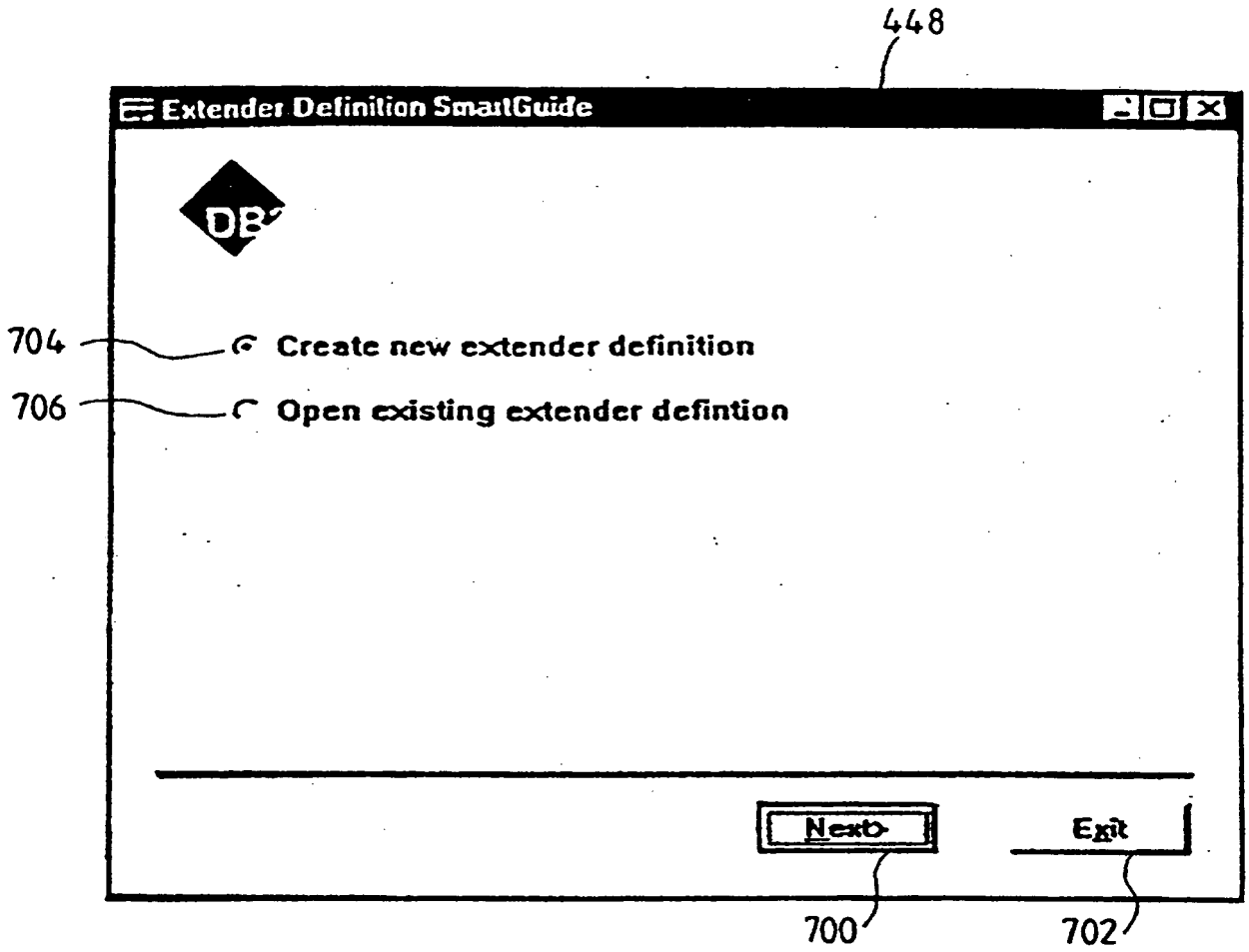
448
Extender Definition SmartGuide
704
Create new extender definition
706
Open existing extender defintion
Next>
Exit
700
702

FIG. 7A

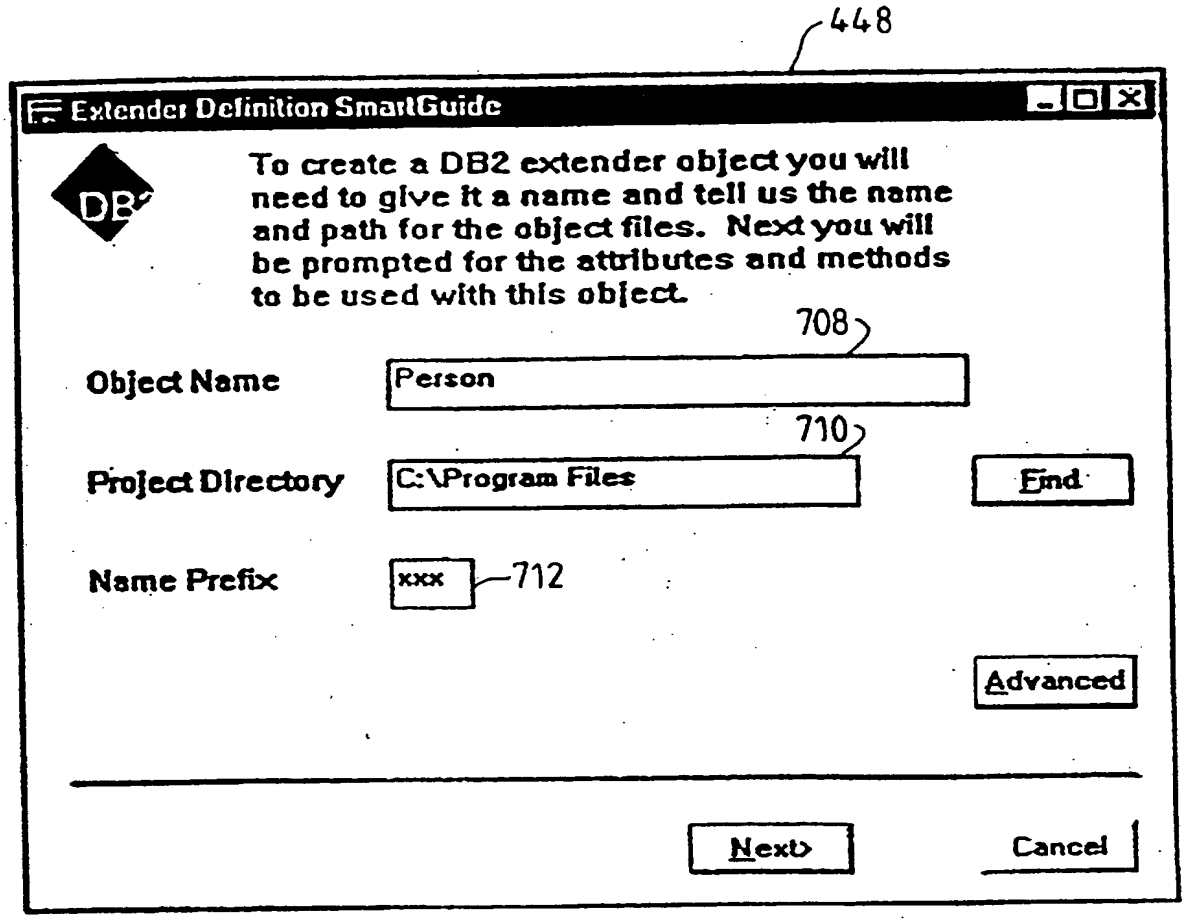
448
Extender Definition SmartGuide
To create a DB2 extender object you will need to give it a name and tell us the name and path for the object files. Next you will be prompted for the attributes and methods to be used with this object.
708
Object Name
Person
710
Project Directory
C:\Program Files
Find
Name Prefix
xxx
712
Advanced
Next>
Cancel

FIG. 7B

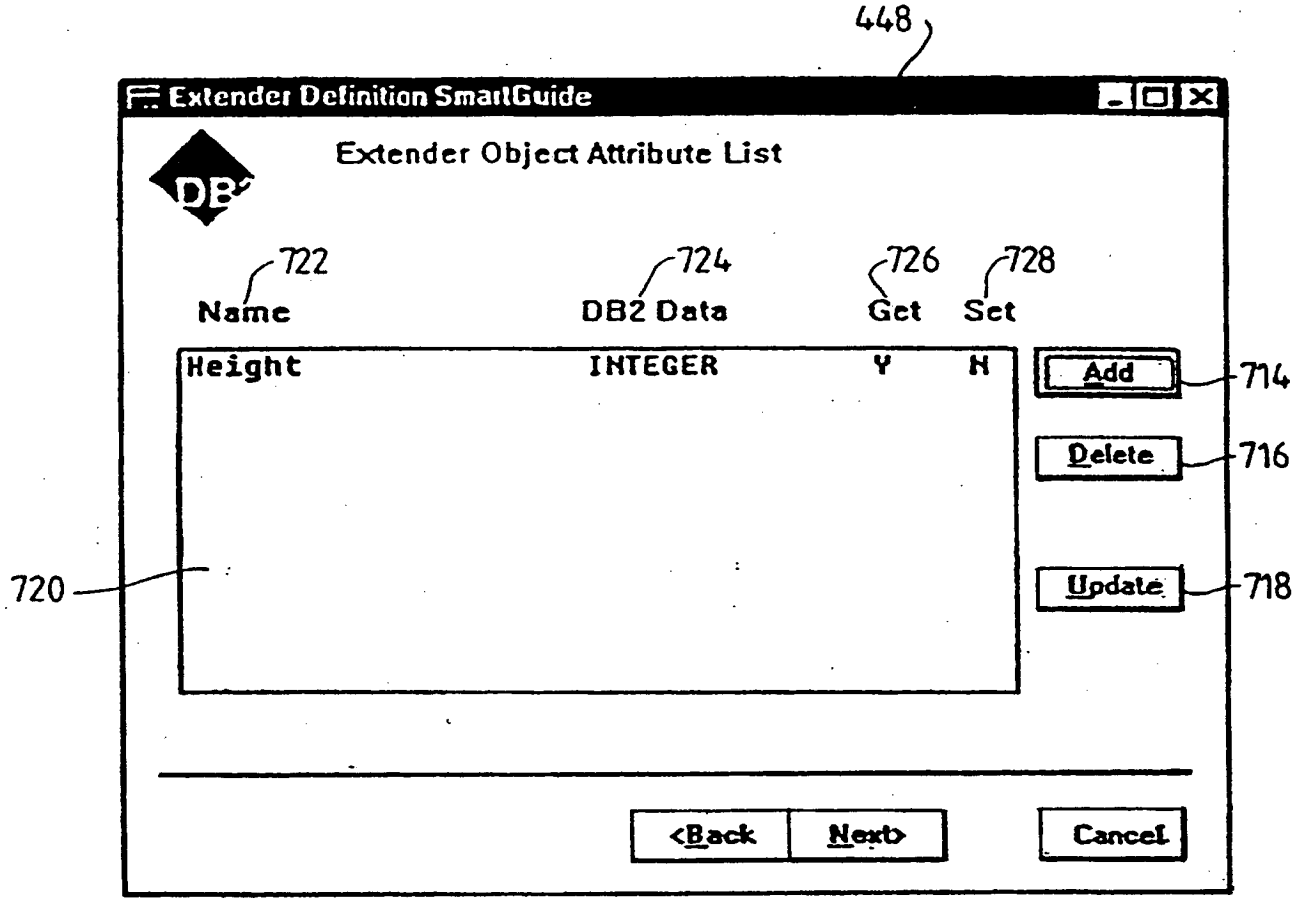
448
Extender Definition SmartGuide
Extender Object Attribute List
722
724
726
728
Name
DB2 Data
Get
Set
Height
INTEGER
Y
N
Add
714
Delete
716
720
Update
718
<Back
Next>
Cancel

FIG. 7C

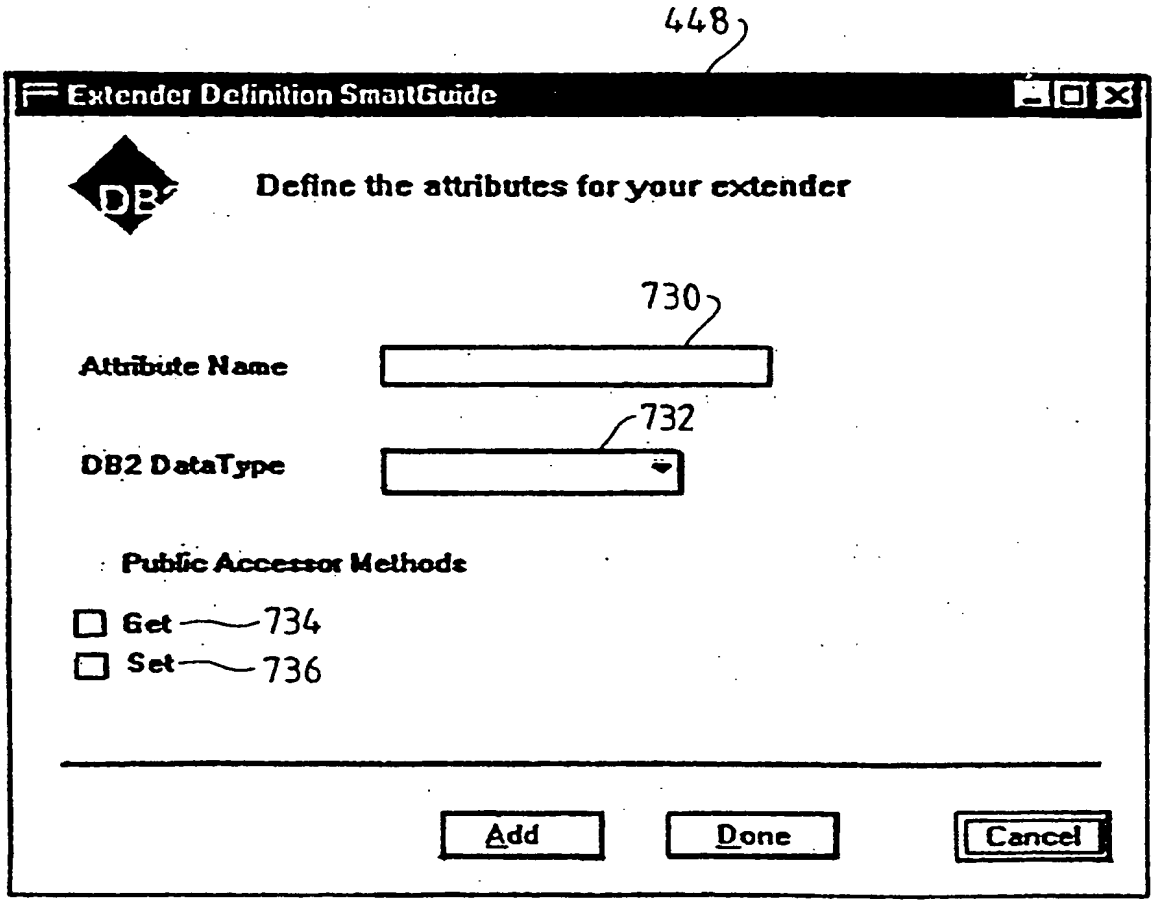
448
Extender Definition SmartGuide
Define the attributes for your extender
730
Attribute Name
732
DB2 DataType
Public Accessor Methods
Get
734
Set
736
Add
Done
Cancel

FIG. 7D

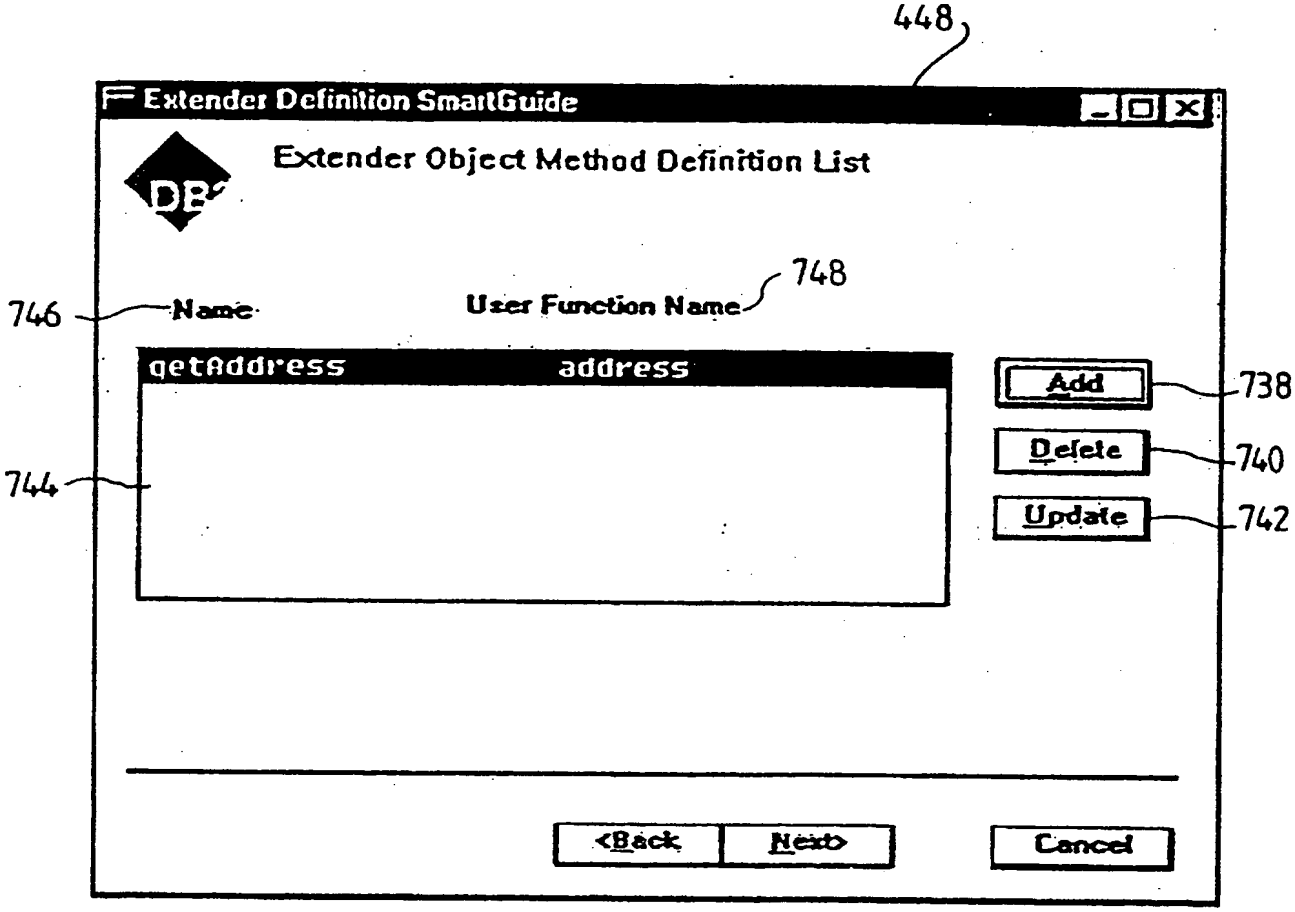
448
Extender Definition SmartGuide
Extender Object Method Definition List
748
746
Name
User Function Name
getAddress
address
744
Add
738
Delete
740
Update
742
<Back
Next>
Cancel

FIG. 7E

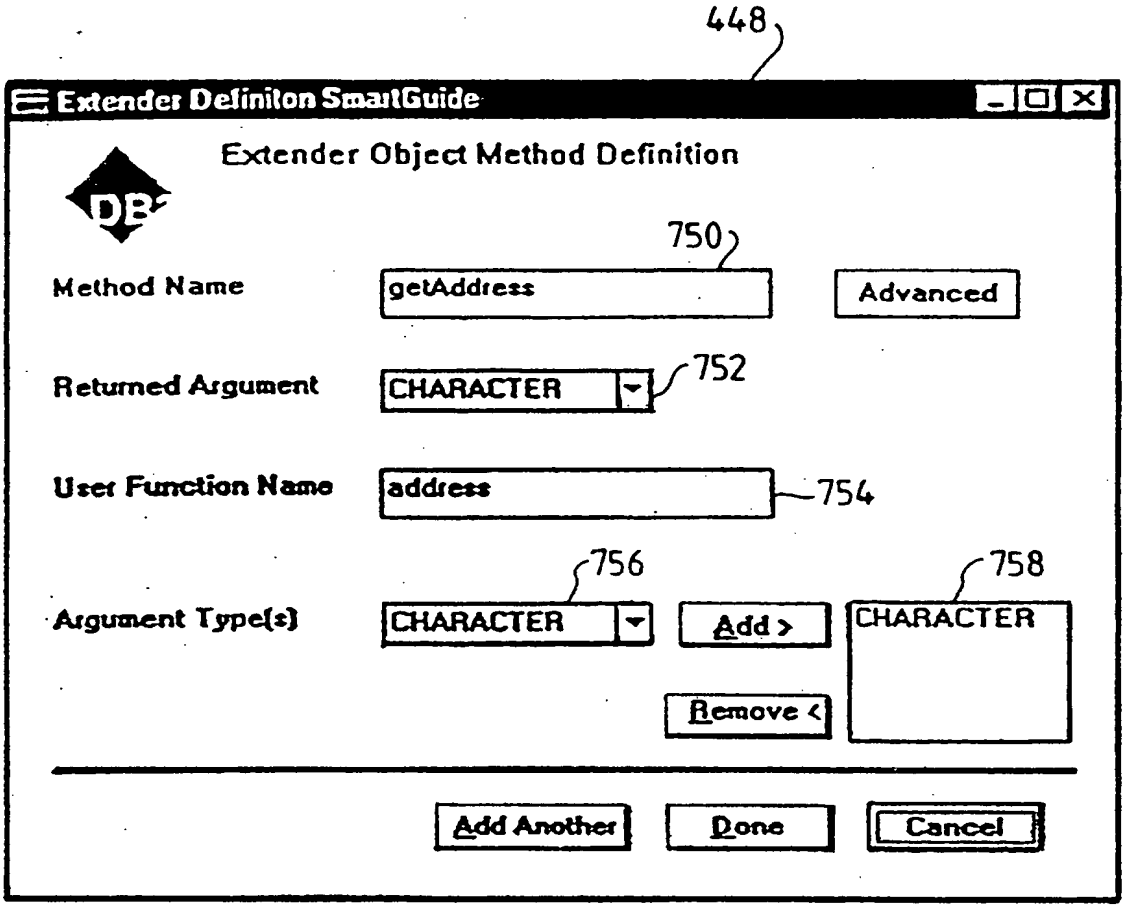
448
Extender Definiton SmartGuide
Extender Object Method Definition
DB2
750
Method Name
getAddress
Advanced
Returned Argument
CHARACTER
752
User Function Name
address
754
756
758
Argument Type(s)
CHARACTER
Add >
CHARACTER
Remove <
Add Another
Done
Cancel

FIG. 7F

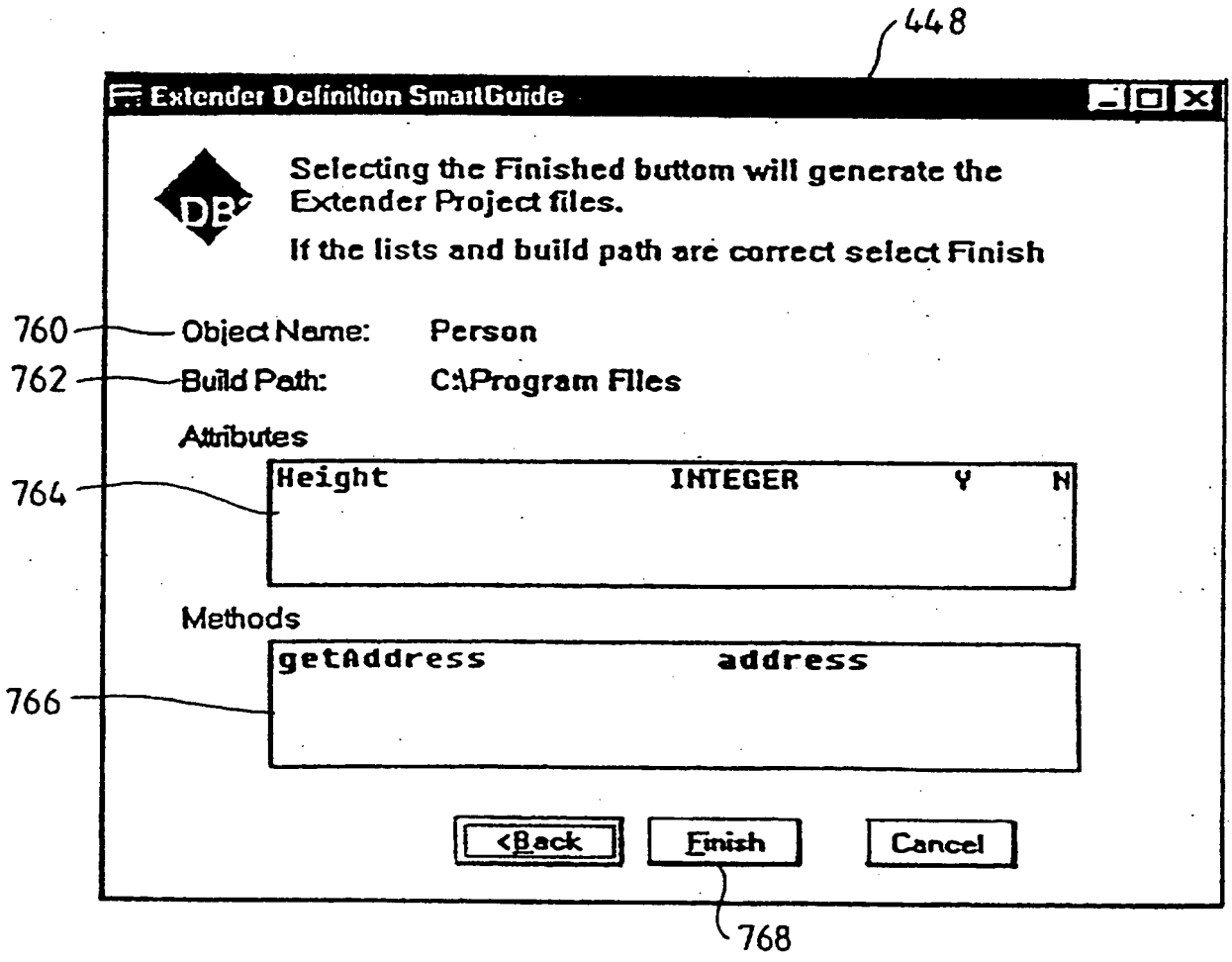
448
Extender Definition SmartGuide
Selecting the Finished buttom will generate the Extender Project files.
If the lists and build path are correct select Finish
760
Object Name:
Person
762
Build Path:
C:\Program Files
Attributes
764
Height INTEGER Y N
Methods
766
getAddress address
<Back
Finish
Cancel
768

FIG. 7G

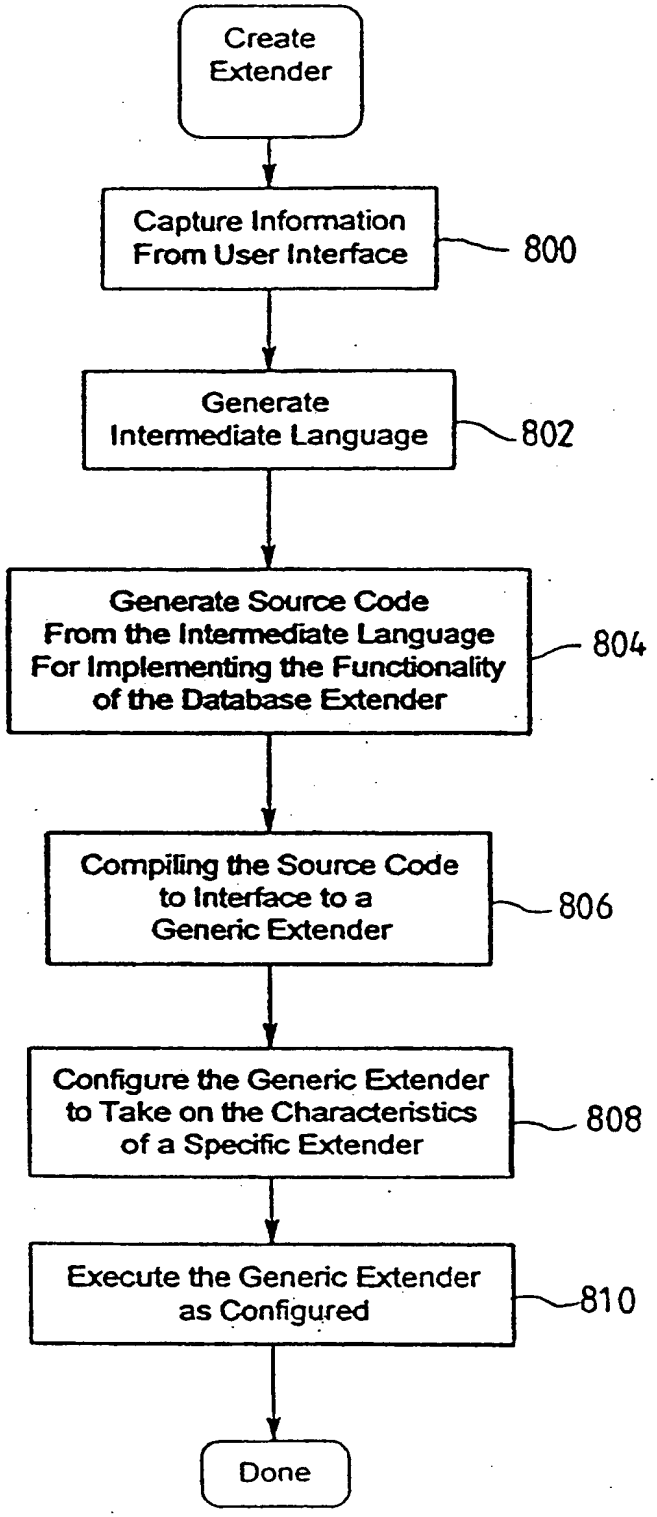
Create Extender
Capture Information From User Interface
800
Generate Intermediate Language
802
Generate Source Code From the Intermediate Language For Implementing the Functionality of the Database Extender
804
Compiling the Source Code to Interface to a Generic Extender
806
Configure the Generic Extender to Take on the Characteristics of a Specific Extender
808
Execute the Generic Extender as Configured
810
Done

FIG. 8

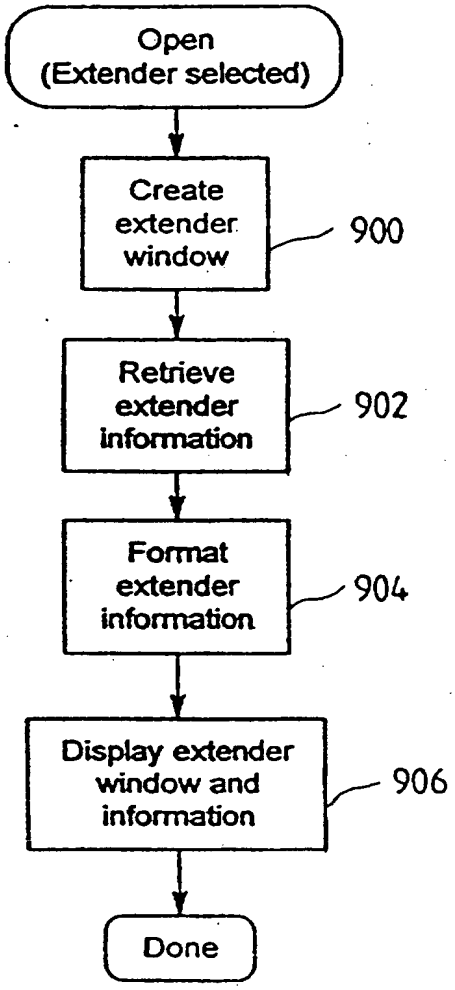
Open
(Extender selected)
Create extender window
900
Retrieve extender information
902
Format extender information
904
Display extender window and information
906
Done

FIG. 9

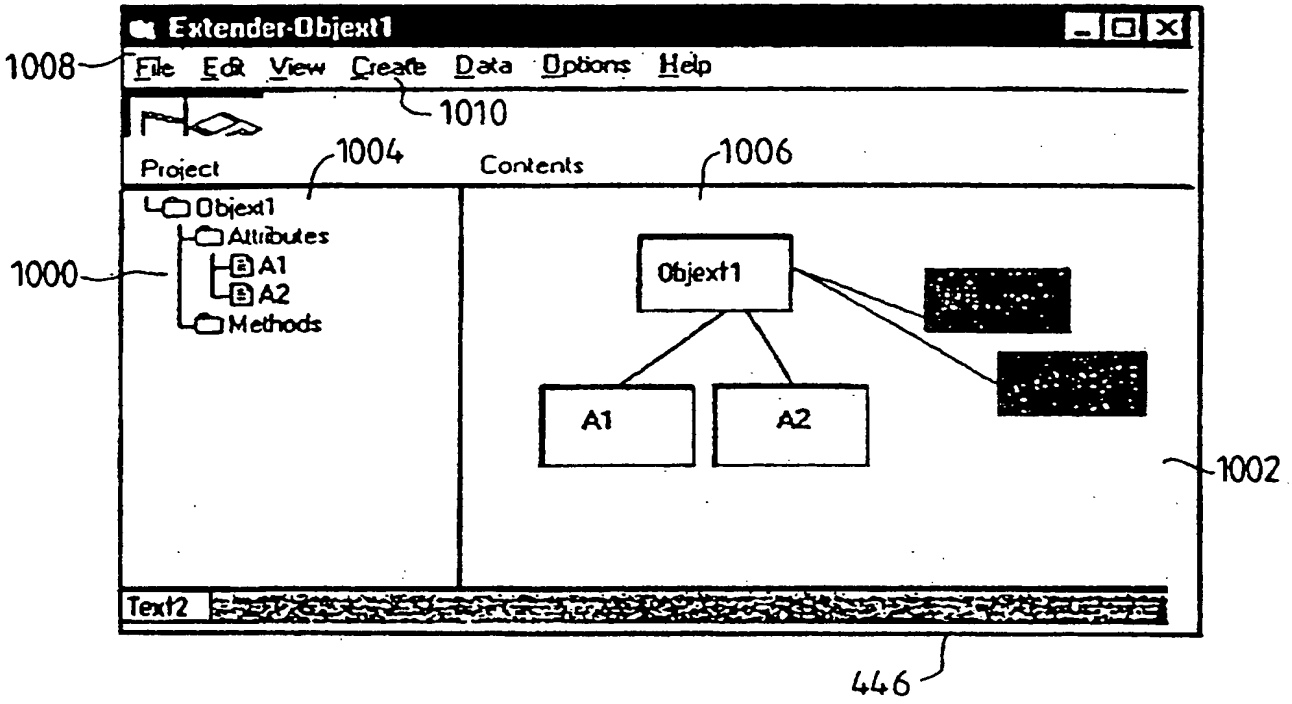
Extender-Object1
File Edit View Create Data Options Help
1008
1010
Project
1004
Contents
1006
Object1
Attributes
A1
A2
Methods
1000
Object1
A1
A2
1002
Text2
446

FIG. 10

This invention relates in general to programming development environments performed by computers, and in particular, to an extender smart guide.

With the fast growing popularity of the Internet and Intranets, especially Web-based networks, there is also a fast growing demand for Internet and Intranet access to databases. However, it is especially difficult to use relational database management system (RDBMS) software with Web-based networks. One of the problems with using RDBMS software with Web-based networks is the lack of programming development environments that can be used to develop both Web-based and RDBMS-based applications.

For example, Web-based networks operate using the HyperText Transfer Protocol (HTTP) and the HyperText Markup Language (HTML). HTTP is the protocol used by Web clients and Web servers to communicate between themselves using these hyperlinks. HTML is the language used by Web servers to create and connect together documents that contain these hyperlinks. This protocol and language results in the communication and display of graphical information that incorporates hyperlinks. Hyperlinks are network addresses that are embedded in a word, phrase, icon or picture that are activated when the user selects a highlighted item displayed in the graphical information.

In contrast, most RDBMS software uses a Structured Query Language (SQL) interface.

The SQL interface has evolved into a standard language for RDBMS software and has been adopted as such by both the American Nationals Standard Organisation (ANSI) and the International Standards Organisation (ISO).

Thus, there is a need in the art for methods of accessing RDBMS software across an Internet or Intranet, and especially via Web-based networks. Further, there is a need for simplified development environments for such systems.

To overcome the limitations in the prior art described above, and to overcome other limitations that will become apparent upon reading and understanding the present specification, the present invention discloses a method, apparatus, and article of manufacture for providing a programming development environment that supports the development of Internet and Intranet applications.

In particular, a database extender can be created using a smart guide. Initially, information from a user interface is captured. An intermediate language is then generated corresponding to the captured information. A source code is generated from the generated intermediate language. The source code implements the functionality of the database extender by interfacing to a generic extender. The interfaced generic extender is configured to take on the characteristics of the specific extender defined in the intermediate language. Then, the configured generic extender is executed as configured to access and maintain the relational integrity of the specific extender.

Referring now to the drawings in which like reference numbers represent corresponding parts throughout:

FIG. 1 is a block diagram that illustrates the three tier architecture of the present invention;

FIG. 2 is a block diagram that further illustrates the components of the network server in the present invention;

FIG. 3 is a block diagram that further illustrates the components of the database server in the present invention;

FIG. 4 is a block diagram that illustrates the development environment of the present invention;

FIG. 5 is a flow chart that illustrates the general logic of the development computer in performing the steps of the present invention, and more specifically, in performing the steps necessary for handling the user interface for the development computer;

FIG. 6 is a flow chart that illustrates the general logic for an open (extender smart guide selected) routine;

FIGS. 7A-7G are diagrams of a computer generated display illustrating the operation of the general logic for an open (extender smart guide selected routine);

FIG. 8 is a flow chart that illustrates the general logic of the extender smart guide in performing the steps of the preferred embodiment of the invention to create an extender;

FIG. 9 is a flow chart that illustrates the general logic for an open (extender selected) routine; and

FIG. 10 is a diagram of a computer generated display illustrating the operation of the general logic for an open (extender selected) routine.

In the following description of the preferred embodiment, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration a specific embodiment in which the invention may be practised. It is to be understood that other embodiments may be utilised and structural and functional changes may be made without departing from the scope of the present invention.

The present invention comprises a computer-implemented Rapid Application Development (RAD) tool for constructing client-server applications for a three tier computer network architecture. The RAD tool provides an Integrated Development Environment (IDE) that is used to design, develop, deploy, and debug computer programming that accesses and displays data quickly and easily on the three tier computer network. Moreover, the RAD tool is extremely easy to use, yet powerful.

The RAD tool of the present invention is primarily targeted to enterprise customers. The fact that an application is produced quickly does not mean that the application is non-critical.

The applications constructed using the RAD tool are primarily oriented towards data access, data manipulation and data rendering, especially in conjunction with relational database management systems (RDBMS).

FIG. 1 is a block diagram that illustrates the three tier architecture 100 of the present invention. Each of the three tiers shown may be executed on separate computer hardware platforms as shown in FIG. 1, or on a single computer hardware platform, or in some combination thereof.

The first tier comprises a client computer 102 having a monitor 104 and one or more data storage devices 106. In the preferred embodiment, the client computer 102 executes a browser 108 capable of containing and executing applets, such as Microsoft Internet Explorer or Netscape Navigator. The browser 108 communicates with programs on other tiers through HTTP (Hypertext Transfer Protocol).

The second tier comprises a network server 110 having one or more data storage devices 112. In the preferred embodiment, the network server 110 executes a plurality of computer programs including a web server 114, a persistent VAB-II runtime module 116, and one or more application scripts 118 retrieved from an APP file 120 stored on a data storage device 112. The web server 114 (such as IBM, Microsoft, or Netscape HTTP daemons) communicates with the browser 108 and the third tier via HTTP.

The VAB-II runtime module 116 executes the application scripts 118 and communicates with the third tier. The application scripts 118 (such as LotusScript scripts) can contain programming logic for communicating with both the browser 108 and the third tier. Preferably, the application scripts 118 include Basic programming instructions, Java, ActiveX, or DLL applet controls, embedded SQL, and other mechanisms known in the art.

The third tier comprises a database server 122 having one or more data storage devices 124 connected thereto. In the preferred embodiment, the database server executes a plurality of computer programs including a relational database management system (RDBMS) 126, a persistent VAB-II runtime module 128, and Stored Procedure (SP) and User Defined Function (UDF) scripts 130 retrieved from an APP file 132 stored on a data storage device 124. The RDBMS 126 (such as IBM's DB2 product) receives requests either directly from tier-2 and/or indirectly from tier-2 via the VAB-II runtime module 128, and then performs the desired database functions. The VAB-II runtime module 128 executes the SP/UDF scripts 130. The SP/UDF scripts 130 comprise programming logic for accessing the database via the RDBMS 126 and communicating with the tier-2 computer programs.

FIG. 2 is a block diagram that further illustrates the components of the network server 110 in the present invention.

The VAB-II runtime module 116, for example, includes an interface 200 to the web server 114, a dispatcher 202, a caching subsystem 204, and a script interpreter 206 for executing one or more application scripts 118 retrieved from one or more APP files 120 stored on a data storage device 112. The interface 200 takes input from the web server 114 via a Common Gateway Interface (CGI), Netscape Server API (NSAPI), Internet Connection Server API (ICAPI), or some other protocol, and converts it to a form for use by the dispatcher 202. The dispatcher 202 then allocates a thread of the VAB-II runtime module 116 to each incoming request to run the desired application script 118. The caching subsystem 204 exists to help manage special purpose object persistence. The script interpreter 206 executes the application script 118 retrieved from the APP file 120 stored on a data storage device 112.

FIG. 3 is a block diagram that further illustrates the components of the database server 122 in the present invention. The VAB-II runtime module 128, for example, includes an interface 300 to the RDBMS 126, a caching subsystem 302, and a script interpreter 304 for executing one or more SP/UDF scripts 130 retrieved from one or more APP files 132 store on the data storage device 124. No dispatcher is required for the VAB-II runtime module 128 in the database server 122. The interface 300 provides a mechanism for invoking the database server 126 from the VAB-II runtime module 128 via a dynamic link library (DLL) or some other protocol.

As in the network server 110, the caching subsystem 302 exists to help manage special purpose object persistence, although SP/UDF scripts 130 are generally not persistent. The script interpreter 304 executes the SP/UDF script 130 retrieved from the APP file 132.

As indicated above, the computer programs of the three tiers shown may be executed on separate computer hardware platforms or on a single computer hardware platform 134 or in some combination thereof. Each of the computers may each include, inter alia, one or more processors, memory, keyboard, or display, and may be connected locally or remotely to fixed and/or removable data storage devices and/or data communications devices. Each of the computers in each of the tiers also could be connected to other computers via the data communications devices.

FIG. 4 is a block diagram that illustrates the development environment of the present invention. A development computer 400 executes a Rapid Application Development (RAD) tool comprised of a number of different computer programs or modules, including a graphical user interface (GUI) 402, project manager 404 and associated builder 406, form editor 408 for constructing HTML forms 410, code editor 412 for constructing scripts 414, debugger 416, SQL painter 418 for constructing queries 420, RDBMS extender guide 422, and RDBMS extender user interface 424, as well as a browser 426, web server 428, VAB-II runtime module 430, and RDBMS 432.

The RAD tool displays a user interface on a monitor 434 attached to the development computer 400, which includes, inter alia, a project window 436, form editor window 438, control pad 440, code editor window 442, debugging window 444, extender user interface window 446, extender guide window 448, SQL painter window 450, as well as a browser window 452.

As described above, the present invention is typically implemented using a plurality of computer programs, each of which executes under the control of an operating system, such as OS/2, Windows, DOS, AIX, UNIX, MVS, etc., and causes the development computer 400 to perform the desired functions as described herein. Thus, using the present specification, the invention may be implemented as a machine, process, or article of manufacture by using standard programming and/or engineering techniques to produce software, firmware, hardware or any combination thereof.

Generally, the computer programs and/or operating system are all tangibly embodied in a computer-readable device or media, such as memory, data storage devices, and/or data communications devices, thereby making a computer program product or article of manufacture according to the invention. As such, the terms "article of manufacture" and "computer program product" as used herein are intended to encompass a computer program accessible from any computer readable device or media.

Moreover, the computer programs and operating system are comprised of instructions which, when read and executed by the development computer 400, causes the computer 400 to perform the steps necessary to implement and/or use the present invention. Under control of the operating system, the computer programs may be loaded from memory, data storage devices, and/or data communications devices into the memory of the development computer 400 for use during actual operations.

Those skilled in the art will recognise many modifications may be made to this configuration without departing from the scope of the present invention. For example, those skilled in the art will recognise that any combination of the above components, or any number of different components, peripherals, and other devices, may be used with the present invention.

RDBMS extenders allow developers to define distinct data types and special functions for many types of non-traditional data. At the highest level, extenders are comprised of a user-defined type (UDT) with one or more user-defined functions (UDFs) which manipulate the UDT, and a set of administrative support tables and triggers which contain meta information about the extender. With the extender approach, complex manipulation of UDTs is implemented once and accessed from many tier-2 applications. This allows quicker development of tier-2 applications and ensures consistency across the applications.

An extender smart guide (or "wizard") 422 is provided by the development environment 400 according to the present invention. The extender smart guide 422 provides a series of "smart guides" or graphical user interfaces 448 for receiving an extender definition (i.e. high level information about the extender) and automatically generates the various programming logic for the extender definition. By providing these smart guides, the extender smart guide 422 allows a developer to build extenders, but the extender smart guide 422 hides the internal mechanisms from the developer.

FIG. 5 is a flow chart that illustrates the general logic of the development computer 400 in performing the steps of the present invention, and more specifically, in performing the steps necessary for handling the user interface for the development computer 400. In the development computer 400, operations are performed when transitions are made, based upon input events, from present or current states to new states.

Block 500 represents the development computer 400 waiting for an input event (e.g., a mouse button click). It should be appreciated that during this time, other system tasks, e.g., file, memory, and video tasks, etc., may also be carried out. When an input event occurs, control passes to block 502 to identify the input event. Based upon the input event, as well as the current state of the development computer 400 determined in block 504, a new state is determined in block 506.

In block 508, a transition is made to the new state and performs any actions required for the transition. In block 510, the current state is set to the previously determined new state, and control returns to block 500 to wait for more input events.

The specific operations that are performed by block 508 when transitioning between states will vary depending upon the current state and the input event. The various operations required to implement the present invention represent particular events handled by the development computer 400. However, it should be appreciated that these events represent merely a subset of all of the events handled by the development computer 400.

FIG. 6 illustrates operations that may be performed by the development computer 400 when an extender smart guide 422 is selected as part of the preferred embodiment of the invention. In FIG. 6, the header block thereof indicates the particular input event with the current state denoted in parentheses. For FIG. 6, corresponding diagrams, i.e., FIG. 7A-7G, are provided to illustrate the operation of the routine.

FIG. 6 is a flow chart that illustrates the general logic for an open (extender smart guide selected) routine. FIGS. 7A-7G are diagrams of a computer generated display 448 illustrating the operation of the routine of FIG. 6.

The open routine opens or creates an extender smart guide window 422 to enable a user to create an extender.

The extender smart guide window 448 is illustrative of a typical graphical user interface (GUI) window, and includes several standard user interface mechanisms, such as that displayed in FIG. 7A, including a title bar (for repositioning), a border (for resizing), a minimise button, a maximise button, and a close button, and may also have scroll bars (for scrolling). Additionally, the user interface 448 contains a next button 700 for advancing to the next smart guide and an exit button 702 for exiting from the extender smart guide window 448. The user interface 448 may also contain a cancel button for returning to a previous smart guide.

In the preferred embodiment, the open routine is executed whenever an "open" event (e.g., a double-clicked mouse button) is recorded after an extender smart guide has been selected. Selection of an extender smart guide may be made in several known manners, e.g., by using various mouse/keyboard combinations to select or highlight the extender smart guide represented visually on monitor 434 of the development computer 400. The act of opening or creating an extender smart guide 422 may be initiated by other functions, e.g., selecting the function in a fixed or pop-up menu.

Block 600 represents the development computer 400 creating the extender smart guide window 448.

The extender smart guide window 448 includes typical user interface mechanisms, such as those displayed in FIG. 7A. A contents area is defined within the extender smart guide window 448.

Block 602 represents the development computer 400 retrieving the extender smart guide information from an APP file stored in a data storage device connected (either locally or remotely) to the development computer 400.

Block 604 represents the development computer 400 formatting the retrieved extender smart guide information for display within the contents area of the extender smart guide window 448.

Block 606 represents the development computer 400 displaying the extender smart guide window 448 and the formatted extender smart guide information on a monitor 434, as shown in FIG. 4. Opening or creation of the extender smart guide window 448 is then complete.

Once an extender smart guide window 448 is created and displayed, various operations are preferably provided to manipulate the extender smart guide window 448 and its contents, as is well known in the art.

FIGS. 7A-7G illustrate smart guide windows 448 provided by the extender smart guide 422 to enable a user to create or modify an extender definition.

Each of these figures includes standard user interface mechanisms, such as a title bar (for repositioning), a border (for resizing), a minimise button, a maximise button and a close button, and may also have scroll bars (for scrolling). Additionally, each user interface typically contains a next button for advancing to the next smart guide and a cancel button for returning to a previous smart guide or an exit button for exiting from the extender smart guide window 448.

FIG. 7A illustrates the first user interface provided by the extender smart guide 422 and displayed in the extender smart guide window 448. This smart guide provides selection boxes 704 and 706 that enable a user to either create a new extender definition or open an existing extender definition. When selection box 704 is selected, the smart guide window 448 illustrated in FIG. 7B is displayed.

FIG. 7B includes an object name text box 708, a project directory text box 710, and a name prefix text box 712 for obtaining information for the extender to be created. Once this information is entered into text boxes 708, 710, and 712, the next smart guide, illustrated in FIG. 7C, is displayed within the smart guide window 448.

FIG. 7C illustrates a smart guide window 448 that is used to obtain attribute information for the extender object that is being created. This smart guide window 448 includes an add button 714 for adding attributes, a delete button 716 for deleting attributes, and an update button 718 for updating attributes.

The list box 720 contains a list of attributes. Each row in the list box 720 contains a name column 722, a DB2 data column 724, a get column 726, and a set column 728, which provide information about each attribute in the list box 720. When the add button 714 or the update button 718 are selected, the next smart guide window 448 displayed enables defining attributes.

FIG. 7D illustrates the smart guide window 448 used to define attributes. An attribute name text box 730 enables entry or update of an attribute name. A DB2 data type text box 732 enables selection of a data type from a drop down list box. The get checkbox 734 and the set checkbox 736 enable selection of get and set. Once attributes have been defined for the extender object, the next smart guide window 448 enables defining methods for the extender object.

FIG. 7E illustrates a smart guide window 448 that is used to obtain a method definition list for the extender object that is being created. This smart guide window 448 includes an add button 738 for adding methods, a delete button 740 for deleting methods, and an update button 742 for updating methods. The list box 744 contains a list of methods. Each row in the list box 744 contains a name column 746 for the method and a user function name column 748 for the user function name of the method. When the add button 738 or the update button 740 are selected, the next smart guide displayed enables defining methods.

FIG. 7F illustrates the smart guide used to define methods. A method name text box 750 enables entry or update of a method name. A returned argument text box 752 enables selection of a data type from a drop down list box for the argument returned by the method. The user function name text box 754 enables entry of the user function name. The argument type text box 756 enables selection of one or more arguments for the method from a drop down list box. These arguments are displayed in list box 758. Once methods have been defined for the extender object, the next smart guide window 448 enables generation of extender project files.

FIG. 7G illustrates a smart guide window 448 for generating extender project files. This smart guide window 448 displays the extender object name 760, a build path 762, an attributes list box 764 containing attributes, and a methods list box 766 containing methods. If the information provided by the smart guide is correct, a user selects the finish button 768, and then the extender smart guide 422 automatically generates the extender project files.

FIG. 8 is a flow chart that illustrates the general logic of the extender smart guide 422 in performing the steps of the preferred embodiment of the invention to create an extender.

Block 800 represents the extender smart guide 422 capturing information from the user interface.

Block 802 represents the extender smart guide 422 generating an intermediate language corresponding to the captured information.

Block 804 represents the extender smart guide 422 generating source code from the intermediate language for implementing the functionality of the database extender. The source code includes user-defined functions for a specific extender, administrative application programming interfaces for enabling and manipulating the specific extender, and an administrative command program that uses the administrative application programming interfaces.

Block 806 represents the extender smart guide 422 compiling the source code to interface to a generic extender.

Block 808 represents the extender smart guide 422 configuring the generic extender to take on the characteristics of a specific extender.

Block 810 illustrates the extender smart guide 422 executing the generic extender as configured to access and maintain the relational integrity of the specific extender.

In an alternative embodiment, the RDBMS 126 can execute a dynamic link library associated with the specific extender, and then the dynamic link library executes the generic extender.

In yet another alternative embodiment, a client browser 102 invokes the specific extender, which in turn invokes the generic extender, which in turn invokes the RDBMS 126.

The programming logic generated by the extender smart guide 422 is divided into two sets of generated code. The first set of generated code includes administrative functions and stored procedures which will be run by the RDBMS 126 to enable a database table to include a column for an extender. This first set of generated code includes defining UDTs and UDFs to the database as well as building the administrative support tables and triggers. The second set of generated code includes boilerplate portions of the UDFs. This second set of generated code contains calls to user supplied modules which contain the actual UDF logic.

The user interface, generator, and the generic extender can be packaged together into a distribution package, however, one skilled in the art would recognise that they can be packaged in other ways.

Additionally, if a user wants to change an extender smart guide interface, the user is able to restart a session with the extender definition originally entered, make modifications, and then the extender smart guide will automatically regenerate the appropriate code.

The development computer 400 also includes an extender 424 which displays an extender window 446 on the monitor 434. The extender window 446 displays a project view of the extender 424 and allows the definition of attributes and access methods for database extender objects. By providing this project view, the extender 424 allows a developer to build extenders, but the extender 424 hides the internal mechanisms from the developer.

FIG. 9 illustrates operations that may be performed by the development computer 400 when an extender 424 is selected as part of the preferred embodiment of the invention. In FIG. 9, the header block thereof indicates the particular input event with the current state denoted in parentheses. For FIG. 9, a corresponding diagram, i.e., FIG. 10, is provided to illustrate the operation of the routine.

FIG. 9 is a flow chart that illustrates the general logic for an open (extender selected) routine. FIG. 10 is a diagram of a computer generated display illustrating the operation of the routine of FIG. 9. The open routine opens or creates an extender window 446 to enable a user to define attributes and access methods for database extender objects.

The extender window 446 is illustrative of a typical graphical user interface (GUI) window, and includes several standard user interface mechanisms, such as that displayed in FIG. 10, including a title bar (for repositioning), a border (for

resizing), a minimise button, a maximise button, and a close button, and may also have scroll bars (for scrolling).

In the preferred embodiment, the open routine is executed whenever an "open" event (e.g., a double-clicked mouse button) is recorded after an extender has been selected. Selection of an extender may be made in several known manners, e.g., by using various mouse/keyboard combinations to select or highlight the extender represented visually on monitor 434. The act of opening or creating an extender may be initiated by other functions, e.g., selecting the function in a fixed or pop-up menu.

Block 900 represents the development computer 400 creating the extender window 446. The extender window 446 includes typical user interface mechanisms, such as those displayed in FIG. 10. A contents area is defined within the extender window 446.

Block 902 represents the development computer 400 retrieving the extender information from an APP file stored in a data storage device connected (either locally or remotely) to the development computer 400.

Block 904 represents the development computer 400 formatting the retrieved extender information for display within the contents area of the extender window 446.

Block 906 represents the development computer 400 displaying the extender window 446 and the formatted extender information on a monitor 434, as shown in FIG. 4. Opening or creation of the extender window 446 is then complete.

Once an extender window 446 is created and displayed, various operations are preferably provided to manipulate the extender window 446 and its contents, as is well known in the art.

FIG. 10 illustrates a relational database extender 424 displayed within the extender window 446. The relational database extender 424 has multiple extender objects. The display includes a project area 1000 and a contents area 1002. The project area 1000 provides a directory structure 1004 extender objects at the highest level, with attributes and methods of each object at lower levels. The contents area 1002 displays an object view 1006 of the extender objects and their attributes and methods.

The extender objects include UDTs and UDFs. These can also be displayed in the extender window 446. Additional UDTs and UDFs can be added to the extender 424. Moreover, a subclass can be invoked to build a new extender 424 from a displayed extender 424. Also, extender object attribute and method definitions can be modified. A menu bar 1008 may be provided to enable each of these actions to be taken. For example, by selecting the create menu 1010, an extender object can be created.

This concludes the description of the preferred embodiment of the invention. The following describes some alternative embodiments for accomplishing the present invention. For example, any type of computer, such as a mainframe, minicomputer, or personal computer, or computer configuration, such as a timesharing mainframe, local area network, or standalone personal computer, could be used with the present invention.

In summary, the present invention discloses a method, apparatus, and article of manufacture for providing a programming development environment that supports the development of Internet and Intranet applications. In particular, a database extender can be created using smart guides. Initially, information from a user interface is captured. An intermediate language is then generated corresponding to the captured information. A source code is generated from the generated intermediate language. The source code implements the functionality of the database extender by interfacing to a generic extender. The interfaced generic extender is configured to take on the characteristics of the specific extender defined in the intermediate language. Then, the configured generic extender is executed as configured to access and maintain the relational integrity of the specific extender.

The foregoing description of the preferred embodiment of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed.

Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto.

CLAIMS

1. A method for creating a database extender comprising the steps of:

capturing information (800) from a user interface;

generating an intermediate language (802) corresponding to the captured information (800);

generating a source code (804) from the generated intermediate language wherein the source code implements the functionality of the database extender by interfacing to a generic extender;

configuring the interfaced generic extender (808) to take on the characteristics of the specific extender defined in the intermediate language; and

executing the configured generic extender (810) as configured to access and maintain relational integrity of the specific extender.

2. An apparatus for creating a database extender, comprising:

a computer (400) having a memory and a data storage device coupled thereto, wherein the data storage device stores a relational database;

one or more computer programs, performed by the computer (400), for capturing information from a user interface, generating an intermediate language (802) corresponding to the captured information, generating a source code (804) from the generated intermediate language wherein the source code implements the functionality of the database extender by interfacing to a generic extender, configuring the interfaced generic extender (808) to take on the characteristics of the specific extender defined in the intermediate language, and executing the

configured generic extender (810) as configured to access and maintain relational integrity of the specific extender.

3. An article of manufacture comprising a program storage medium readable by a computer (400) and embodying one or more instructions executable by the computer to perform method steps for creating a database extender, the database extender being created by the computer to retrieve data from a database stored in an data storage device coupled to the computer, the method comprising the steps of:

capturing information (800) from a user interface;

generating an intermediate language (802) corresponding to the captured information;

generating a source code (804) from the generated intermediate language wherein the source code implements the functionality of the database extender by interfacing to a generic extender;

configuring the interfaced generic extender (808) to take on the characteristics of the specific extender defined in the intermediate language; and

executing the configured generic extender (810) as configured to access and maintain relational integrity of the specific extender.

4. The method of claim 1 or apparatus of claim 2 or article of claim 3, wherein a generator performs the steps of generating, further comprising the step or means of packaging the user interface (402), the generator and the generic extender into a distribution package.

5. The method of claim 1 or the apparatus of claim 2 or the article of claim 3, wherein the source code includes user-defined functions for a specific extender, administrative application programming interfaces for enabling and manipulating the specific extender, and an administrative command program that uses the administrative application programming interfaces.

6. The method apparatus or article of claim 5, further comprising the step or means of compiling the source code (806) to interface to the generic extender.

7. The method of claim 1 or apparatus of claim 2 or article of claim 3, wherein a dynamic link library is associated with the specific extender, wherein a relational database management system (RDBMS) executes the dynamic link library, and wherein the dynamic link library executes the generic extender.

8. The method of claim 1 or apparatus of claim 2 or article of claim 3, wherein a client application invokes the specific extender, wherein the specific extender invokes the generic extender, and wherein the generic extender invokes the relational database management system.

The Patent Office

29

Application No: GB 9806003.1
Claims searched: 1-8

Examiner: K. Sylvan
Date of search: 24 September 1998

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.P): G4A (AUDB)

Int Cl (Ed.6): G06F (17/30)

Other: Online: WPI,COMPUTER

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
	None	

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.